ソフトゼミＡ　第3回　for 文，while 文

はじめに

今回は for 文や while 文を使ったプログラムの『**繰り返し**』について学びましょう。

例えば「(」・ω・)」うー！(／・ω・)／にゃー！」と 100 回入力したくなったとします。このとき　printf(“(」・ω・)」うー！(／・ω・)／にゃー！”);　とキーボードで打ち、コピー&ペーストしてもよいですが、ちょっとメンドクサイですよね。そこで出てくるのが、プログラムを繰り返し処理する命令の for 文と while 文です。

for 文

- for 文の基礎

まずは for 文について説明していきます。for 文を使った繰り返しは基本的に次のように書かれます。

a03_1.c

```
#include <stdio.h>
int main(void){
        int i;
        for(i=0;i<5;i++){
                printf(“(」・ω・)」うー！(／・ω・)／にゃー！”);
        }
        printf(“¥n”);
        return(0);
}
```

注目してほしいのは for の後ろの()の中身です。数式みたいなのが３つ、セミコロン(;)で区切って書いてあるのが分かると思います。これらは初期化・制御式・後処理といい、for 文の構成の基礎となる３つの記述です。↓のような感じです。

```
for(  i=0 ; i<5 ; i++ ){
```

初期化　制御式　後処理

・初期化…繰り返しの前に一度だけ実行されます。今は変数 i に 0 を代入して i の初期値を 0 としています。
・制御式…for 文を繰り返し続ける条件が書かれます。この場合は i が 5 より小さいうちは for 文内の処理を繰り返し、5 以上になったら処理をやめます。
・後処理…for 文内の式が実行された後の処理です。i++は i=i+1 と同じ意味で、for 文によって繰り返しが起こるたびに i が 1 ずつ増えていきます。

i のように繰り返し(ループ)の回数を数える変数をループ変数と呼びます。a04_1.c の for 文を言葉で表すと『i を 0 から数えて 5 になるまで繰り返す』となります。i の上限を変えてやれば 100 回だろうが 1000 回だろうが同じ文が楽々書けます。便利ですね。

また、5 回のループを書くなら for(i=1;i<=5;i++) でもいいの？という疑問のお持ちの方もいるかと思います。文法上どちらで書いても大丈夫なのですが、後に『配列』を使うときに for(i=0;i<5;i++) のような 0 から始める書き方に慣れておくと少し楽できます。

- for 文の応用

for 文の基礎で初期化・制御式・後処理について説明しましたが、これらは自由に省略することができます。for(;i<100;i++) for(i=0; ;i++) for(i=1;i<10;) for(; ;i++) for(;;) など、全て for 文として正しい書き方です。ただしセミコロン(;)は省略してはいけないので注意です。

これを使うことで無限ループを書くこともできます。

a03_2.c

```
#include <stdio.h>
int main(void){
        int input;
        /*無限ループ*/
        for(;;){
                printf(“100 を入力してください¥n”);
                scanf(“%d”,&input);
                if(input==100){break;}
        }
        return(0);
}
```

a04_2.c のプログラムは 100 が入力されるまで無限ループします。このとき for(;;)は初期化せず、無条件に繰り返し、後処理もしないという処理をしています。また、for 文内の if 文の中に break と書いてあります。break はループから強制的に抜け出す命令であり、今回は変数 input＝100 という条件が満たされた場合に無限ループを抜けられるようになっています。無限ループを作るときは必ずループから抜ける条件を書いておきましょう。

for 文を 2 重に書くことで、より高度な処理を行うこともできます。

a03_3.c

```
#include<stdio.h>
int main(void){
  /*ループ変数*/
  int i,j;

  /*3回　(」・ω・)」うー！(／・ω・)／にゃー！　するごとに改行*/
  for(i=0;i<10;i++){
        for(j=0;j<3;j++){
              printf(“(」・ω・)」うー！(／・ω・)／にゃー！”);
        }
        printf(“¥n”);
  }
  return(0);
}
```

(」・ω・)」うー！(／・ω・)／にゃー！ を 1 行に 3 回ずつ、10 行書いてみました。変数 i でのループが 1 回繰り返される間に変数 j のループが 3 回繰り返されるようになっています。つまり、i のループの処理が 10 回、j のループの処理が 30 回行われています。

注）実際にはコマンドプロンプトの横幅が足りないので、1 行に 3 つ()・ω・)」うー！(／・ω・)／にゃー！は収まりません。

for 文の説明はこ↑こ↓で終わりです。お疲れ様でした！

次のページから while 文の説明になります。あと少し、頑張りましょう！

while 文と do-while 文

- while 文

  while 文を使ったプログラムは次のように書きます。出力結果は a03_1.c と同じものになります。

  a03_4.c

```
#include<stdio.h>
int main(void){
    /*ループ変数*/
    int i=0;
    while(i<5){
            printf(“(」・ω・)」うー！(／・ω・)／にゃー！”);
            i++;
    }
    printf(“¥n”);
    return(0);
}
```

for 文と似た書き方がされていますね。while 文の()の中は for 文においての制御式のみが書かれます。初期化は while 文の前で行う必要があり、後処理は while 文の内部に書きます。

- do-while 文

do-while 文を使ったプログラムは次のようになります。これも同じ出力結果になります。

a03_5.c

```
#include<stdio.h>
int main(void){
      /*ループ変数*/
      int i=0;
      do{
            printf(“(」・ω・)」うー！(／・ω・)／にゃー！”);
            i++;
      } while(i<5);
      printf(“¥n”);
      return(0);
}
```

do の{}に囲われていますが、**while(i<5)の後にセミコロンがある**のが特徴です。

それぞれの特徴

今回の講習では色々な繰り返しの書き方が出てきました。どれ使えばいいの？ってなる人もいると思うので、それぞれの特徴をまとめてみます。

- **for 文**
  for 文の良いところは条件、初期化が見やすいところです。基本的には for 文を使うと良いでしょう。
- **while 文**
  while 文の良いところはループから抜ける条件が見やすいところです。脱出条件が複雑な繰り返しや無限ループは while 文で書くと良いでしょう。
- **do-while 文**
  do-while 文は中身が少なくとも一度は実行されます。そういった場合にのみ使うと思って大丈夫です。

大体こんな感じです。深く考えずに、プログラムを書いて慣れてみましょう！

■　練習問題

次のプログラムを繰り返しで作り、実行しなさい。

1. (」・ω・)」うー！(／・ω・)／にゃー！を 10 行×10 列で書くプログラム

注）やっぱりコマンド プロンプトの端まで行ってしまうので、下記のようにして出力結果をテキストファイルにしてみるといいかもしれません。

例）ソースコードが a03_q1.c、出力するファイルにつけたい名前が t.txt の場合、コマンド プロンプト上で

bcc32 a03_q1.c	←コンパイル
a03_q1 > **t.txt**	←実行(出力を t.txt)に変更

としてから､テキストエディタで t.txt を開くと 10×10 に並んでいることを確認できます。

2. 5 の倍数が入力されるまでループするプログラム
3. 2 重ループで九九を表示するプログラム